Amendment(s) to the Specification

Please amend the TITLE so it recites the following:

INTREPRETER FOR EXECUTING COMPUTER PROGRAMS AND METHOD FOR COLLECTING STATISTICS

Please replace paragraphs [0013]-[0037] of the SUMMARY OF THE INVENTION section with the following:

[0013] ~~Hence, the present invention provides an interpreter performing operations specified in a computer program consisting of instructions, said instructions being translated to an intermediate format comprising an intermediate code for said instructions, using a service routine to perform the semantics of an instruction. Embodiments of the interpreter thus comprises:~~ The present invention is directed to an interpreter used to facilitate execution of program code that groups service routines into functions based on their frequency of execution using collected frequency of execution statistics. The interpreter preferably does so dynamically and can be implemented in hardware or software.

[0014] ~~statistics means for collecting and recording or generating statistics of how often service routines are executed and what parameters they had;~~ Program code, in the form of a computer program, includes instructions performed by the interpreter via corresponding service routines to perform the corresponding task or task defined thereby. The instructions are preferably first translated into an intermediate format comprising an intermediate code associated therewith.

[0015] ~~clustering means for grouping frequently used service routines with program jumps between each other in a program function with regard to a predetermined frequency value for determining such service routines;~~ The interpreter is configured to keep track of how often service routines are executed along with their parameters. The interpreter is configured to use the statistical frequency of service routine execution for the service routines to group frequently used service routines into a particular function using a predetermined frequency value. Where the interpreter has grouped service routines into two or more functions, jumps preferably are designated between each one of the functions. The interpreter preferably is also configured to keep track of how often service routines are executed after those of a particular function and preferably also is configured to assign more frequently executed service routines a shorter code than service routines executed after the aforementioned particular function. The gathered statistics from an execution are used by the interpreter to optimize service routines by grouping them and coding them in this manner resulting in faster execution speed.

[0016] ~~said statistics means recording the frequency of service routines executed after a said function; and~~ Where the interpreter is configured to assign a shorter code to more frequently executed service routines, memory bandwidth usage is reduced when fetching intermediate code because the corresponding instruction is shorter. The interpreter can be configured to collect frequency of execution statistics before any simulator is compiled or can be configured to collect frequency of execution statistics and dynamically update a function by changing which service routines are grouped in it while the simulator is running.

[0017] ~~encoding means for assigning a frequently used service routines a shorter code than service routines executed after a said function, thus, gathered statistics, from an execution, control an encoding to optimize frequently used service routines for faster execution speed.~~ Instruction sets written in a high level programming language can be used for a specific simulator task and translated into service routines capable of being statistically analyzed and assigned execution frequency based codes using the interpreter. The interpreter itself can be written in a high level programming language, such as standard ISO C.

[0018] ~~In one embodiment of the present invention, more than one program function is grouped by grouping frequently used service routines with jumps between each other in the same function, thus minimizing jumps between functions.~~ The interpreter can be configured to share a program branch of a service routine in a particular function with all other service routines in that particular function. The interpreter can be configured to use a branch prediction table to reduce hard-to-predict jumps to reduce the number of jumps causing the table to function better on current processor architectures. The interpreter can also be configured to employ profile driven compilation to further enhance performance. The interpreter can be further configured to employ automatic compiler register mapping of often used variables by allocating them as local variables.

[0019] ~~Another embodiment provides encoding means for a reduced bandwidth to an electronic memory when fetching intermediate code, as a frequently executed instruction is shorter.~~ The interpreter can be configured so it does not use compiler specific extensions thereby providing compiler independence. If desired, the interpreter can be used by an emulator. The interpreter can also be configured to improve instruction cache performance by placing frequently used codes of service routines in the same function in a sequential block.

[0020] ~~A further embodiment comprises that said statistics is collected before a simulator is compiled.~~ In a preferred method of interpreter operation, instructions of a computer program are preferably translated into an intermediate format that comprises an intermediate code such that service routines are used in carrying out program instructions. The interpreter collects and records statistics of how often the service routines are executed along with their parameters. Frequently used service routines are grouped into their own program function based on a predetermined frequency value. There preferably is a program jump designated between each program function to every other program function. There preferably also is a program jump or branch between each service routine to every other service routine within a particular program function. The interpreter preferably also keeps track of how frequently other service routines are executed after a particular program function. In an encoding step, the interpreter assigns more frequently executed service routines a shorter code. Shorter codes are preferably assigned to more frequently executed service routines than the codes assigned to service routines executed after a particular program function. The interpreter is able to do this after each execution to dynamically optimize frequently used service routines for faster execution speed.

[0021] ~~A still further embodiment comprises that said statistics is collected while a simulator is running, in which case, the simulator dynamically updates a function with service routines used to simulate an instruction set.~~ An interpreter constructed in accordance with the invention can be implemented in the form of a computer program that is configured to perform an above described method in accordance with the invention using a data processor or a computer system.

[0022] ~~Yet another embodiment comprises that said statistics and the encoding provides that a realistic instruction set that is translated to service routines can be written in a high-level programming language for a specific simulator task.~~

[0023] ~~In yet one embodiment the interpreter is provided to be written in the standard ISO C through said statistics and encoding.~~

[0024] ~~A still further embodiment encompasses that a program branch to a next service routine may be shared by all service routines in a function through said statistics and encoding.~~

[0025] ~~A further embodiment reduces the number of jumps, which are hard to predict, in a branch prediction table, causing a table to functioning better on current processor architectures.~~

[0026] ~~One other embodiment sets forth that a profile driven compilation can be used to further enhance a simulator performance through said statistics and encoding.~~

[0027] ~~Yet another embodiment provides for automatic compiler register mapping of often used variables, as the variables may be allocated as local variables.~~

[0028] ~~Another embodiment of the present invention provides for avoiding use of compiler specific extensions, thus providing for compiler independence.~~

[0029] ~~A further embodiment provides for improving instruction cache performance by placing frequent codes in a sequential block due to a common function for service routines.~~

[0030] ~~Another embodiment provides that the interpreter is used by an emulator.~~

[0031] ~~The present invention also sets forth a method for an interpreter, performing operations specified in a computer program consisting of instructions, said instructions being translated to an intermediate format comprising an intermediate code for said instructions, using a service routines to perform the semantics of an instruction. The method comprising the steps of:~~

[0032] ~~collecting and recording statistics of how often service routines are executed and what parameters they had;~~

[0033] ~~grouping frequently used service routines with program jumps between each other in a program function with regard to a predetermined frequency value for determining such service routines;~~

[0034] ~~said statistics recording the frequency of service routines executed after a said function; and~~

[0035] ~~encoding for assigning frequently used service routines a shorter code than service routines executed after a said function, thus gathered statistics, from an execution, control an encoding to optimize frequently used service routines for faster execution speed.~~

[0036] ~~The method is also able to set forth embodiments of the interpreter above as described through attached dependent claims.~~

[0037] ~~The invention is conveniently realized as a computer program product comprising computer program code devised to adapt a data processor or a computer system to perform the steps and functions of the inventive method and apparatus.~~